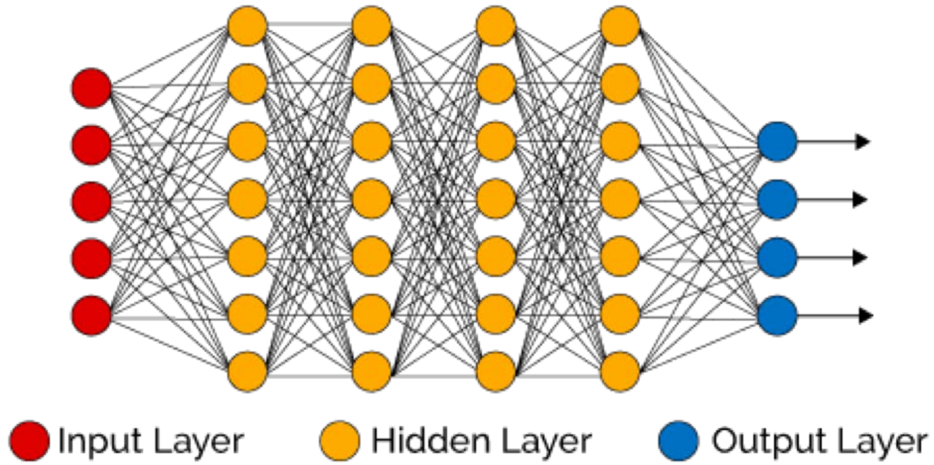
Comparison Between Bare-metal, Container and VM using Tensorflow Image Classification Benchmarks for Deep Learning Cloud Platform

***Chan-Yi Lin**, Hsin-Yu Pai and Jerry Chou
Department of Computer Science, National Tsing Hua University, Taiwan

March 20th, 2018
The 8th International Conference on Cloud Computing and Services Science, CLOSER 2018

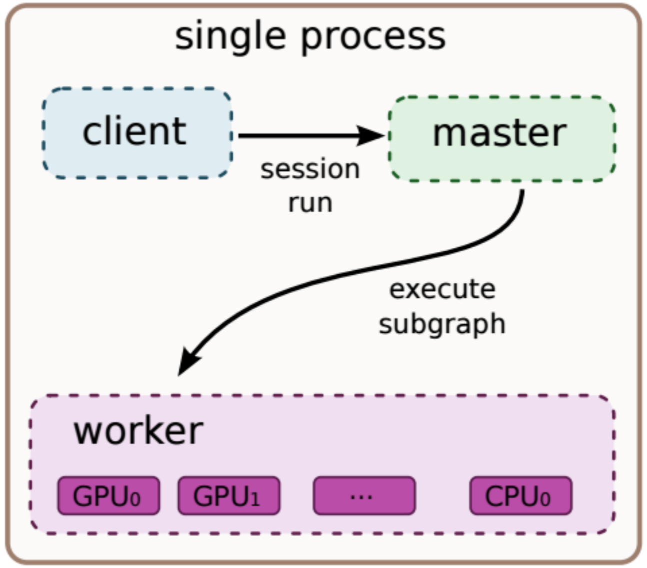
Deep Learning

- A function approximator.
  - **neurons**: receive and process the signals and transmit a signal to another neuron.
  - **layers**: the neurons form numerous layers normally in the scale of 100 to 1000.
  - **weights**: increase or decrease the strength of the signal that it sends downstream. Millions of weights need to be calculated and update during training.
- Train a model by a series of forward and backward propagation to update the weights.
- Use frameworks to easily build a model:
  - Tensorflow, Theano, Torch, etc.

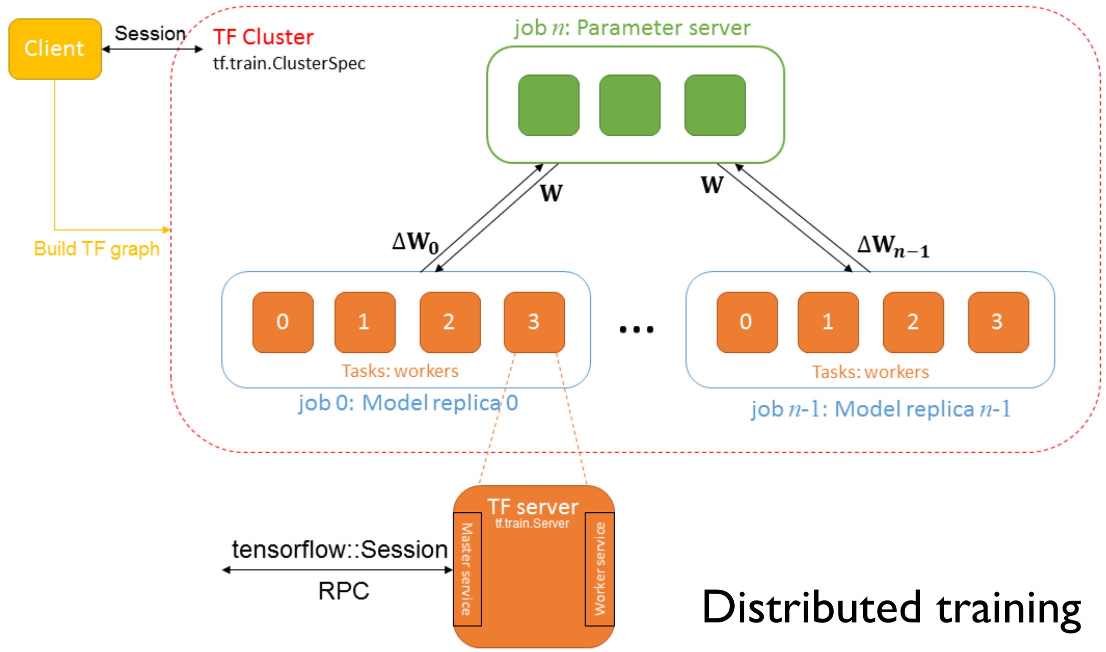


Tensorflow

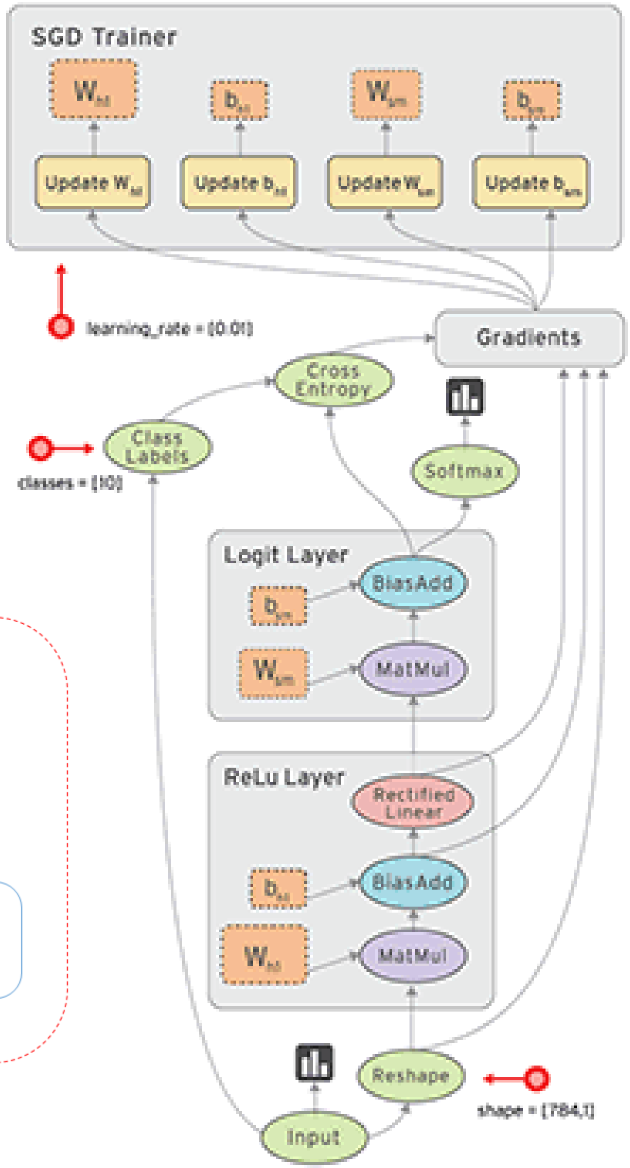
- Data Flow Graphs
- **Single hosts training** and **Distributed training.**



Single host training



Distributed training



Data Flow Graphs

How to Train a Deep Learning Model

<u>On Premises:</u>

- Fully administration control.
- Custom software stack and infrastructure.
- Data privacy.

However,

- Cost of maintenance.
- Not easy to share resources amount users and training tasks. Interference between users.
- Limited by the environments installed.

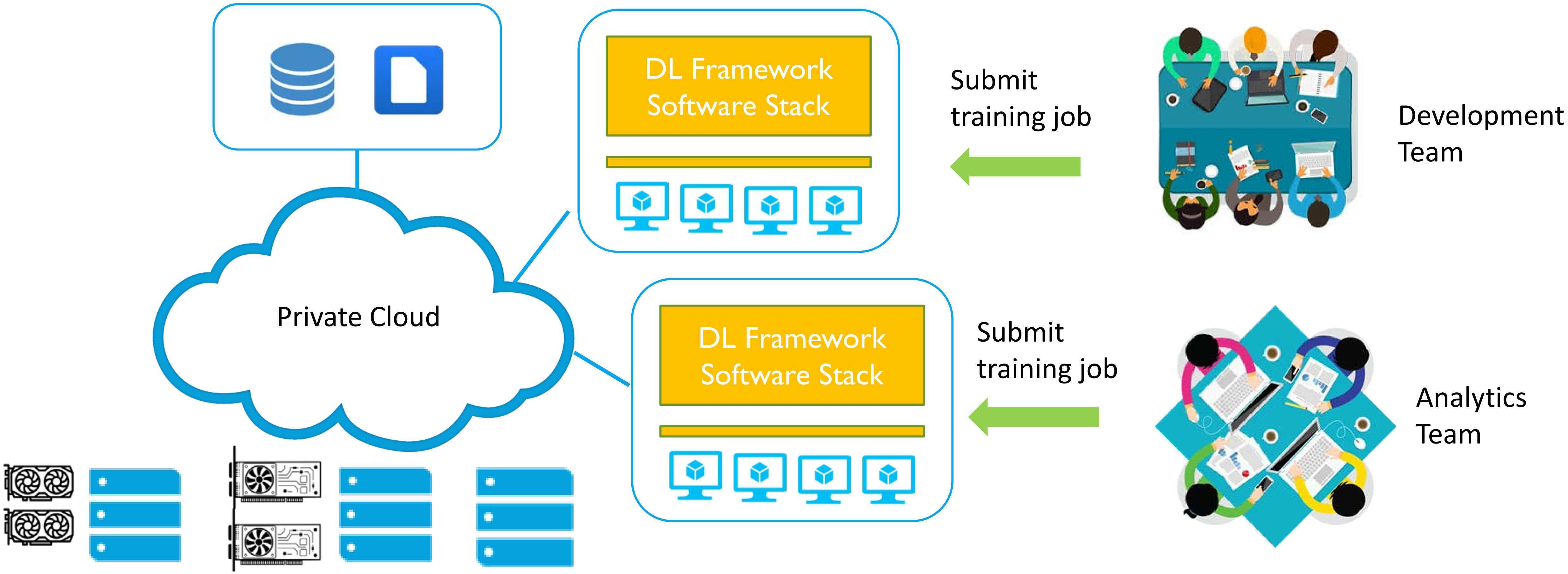
<u>Public Cloud:</u>

- Researchers and developer are freed from the infrastructure.
- Pay-as-you-used.
- Elastic.
- Fault tolerance and resilience

However,

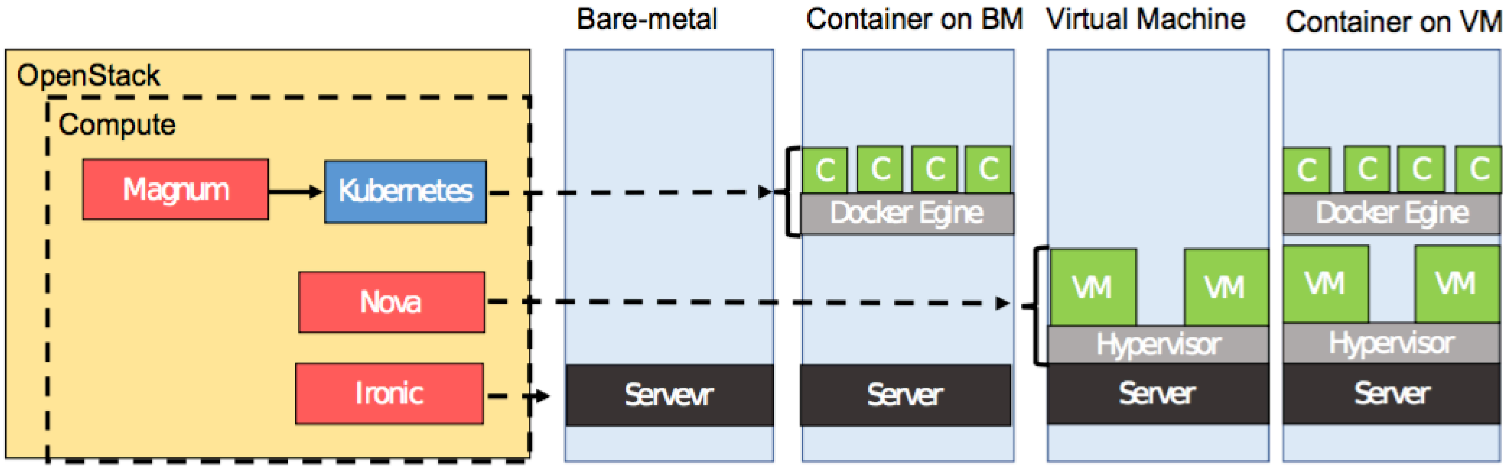
- Very expensive for using instances with GPUs on the public cloud.
- Limited by the cloud providers from both performance and functions perspective.

Private Cloud for Deep Learning



Private Cloud Resource Orchestration

- For building private cloud, **OpenStack** is a well known and open source solution.
- OpenStack offers following types as compute instance:
  - Bare metal: through ***Ironic***
  - Virtual Machine: through ***Nova***
  - Docker Container: through ***Magnum*** and Kubernetes



Private Cloud Resource Orchestration - Related Work

- We knew that the performance ranking is basically
  Baremetal ≅ Container > VM from CPU, I/O and Network perspective.

- However, most of the results are from the tests of using singe resource-bound benchmark applications.

- For deep learning,
  - Complex application with **mixed types of resource usage**.
  - Can have **different settings** and training modes.
  - Can combined with other **orchestration tools**, like Kubernetes.
  - **GPU** involved which means the data transfer in the **I/O bus** should be considered.

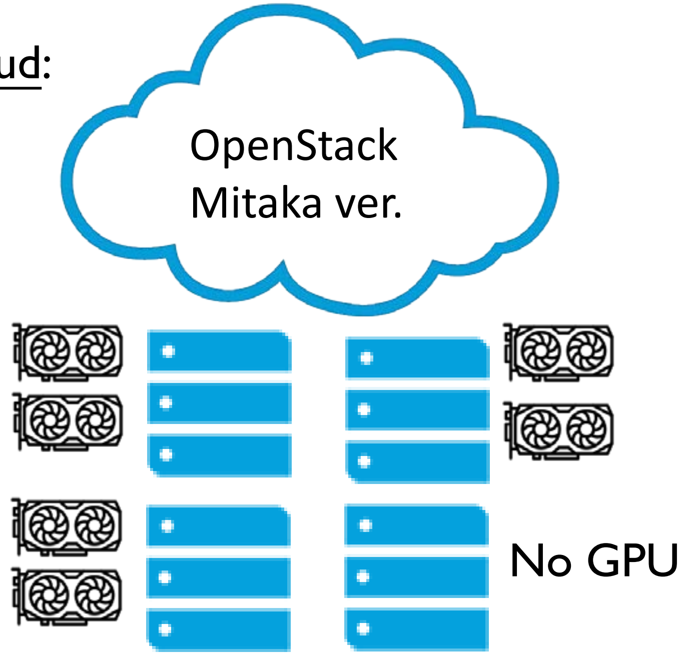
How cloud resources should be orchestrated for deep learning?

Agenda

- Introduction and Background
- Methodology
- Experiment Results
- Conclusion
- Future Work

Environment

Cloud:



CPU: Intel(R) Core(TM) i5-6600 * 4
RAM: 62GB
Network: 1Gbp
GPU: GeForce GTX 1080 *2

Software:

- Tensorflow-gpu: v1.4
- CUDA: v8.0
- cudnn: v6.0
- Docker: v17.05.0-ce
- Kubernetes: v1.7.5, with flannel network overlay.

Use GPU:

- VM: PCI Passthrough.
- Docker Container: use cgroup to map GPU devices onto the container.

Workload

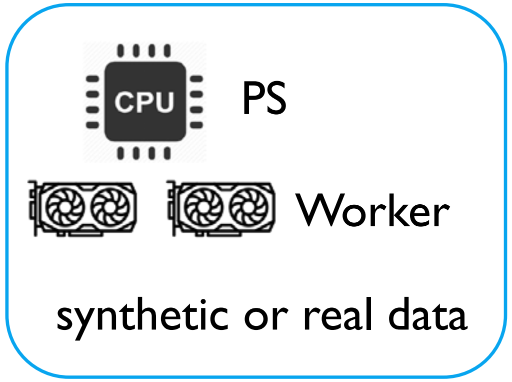
- **Tensorflow** with image classification models as benchmarks.
  - Inception V3, ResNet-50, AlexNet.
  - **Synthetic dataset** and **real dataset (replicate the dataset and place it on every node).**

- A **test training job**: 10 warm-up steps followed by another 100 training steps.

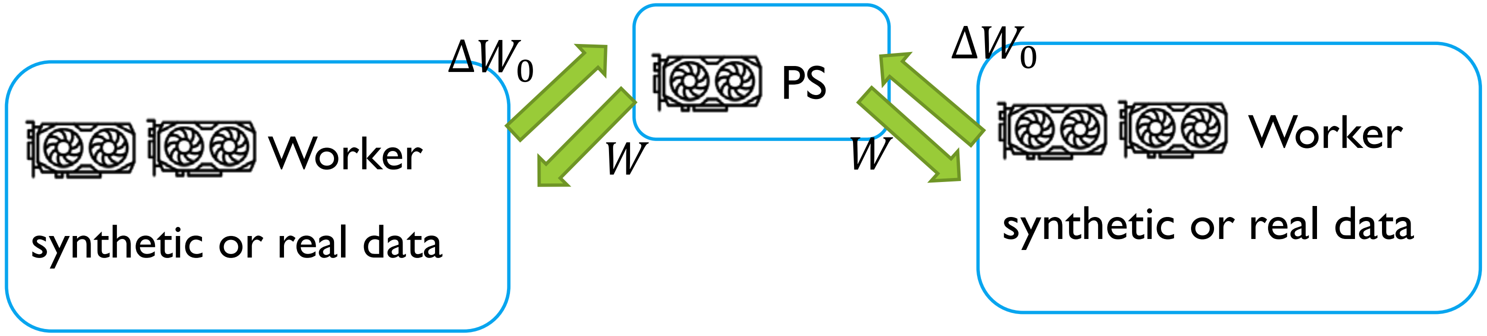
- Two performance metrics.
  - **throughput**: images / sec of each training step.
  - **elapse time**: total execution time from a training job being launched to finished.

Scenarios

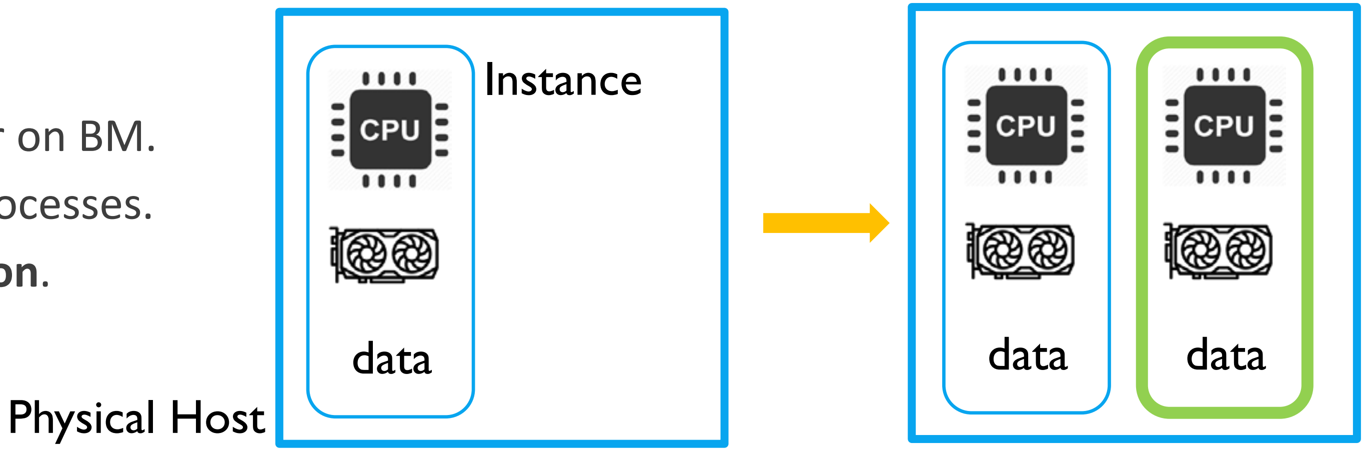
- **Single instance scenario**



- **Distributed multi-instance scenario**



- **Shared resource scenario**
  - Only compare BM, VM and container on BM.
  - For BM, two instances means two processes.
  - Observe the **performance degradation**.

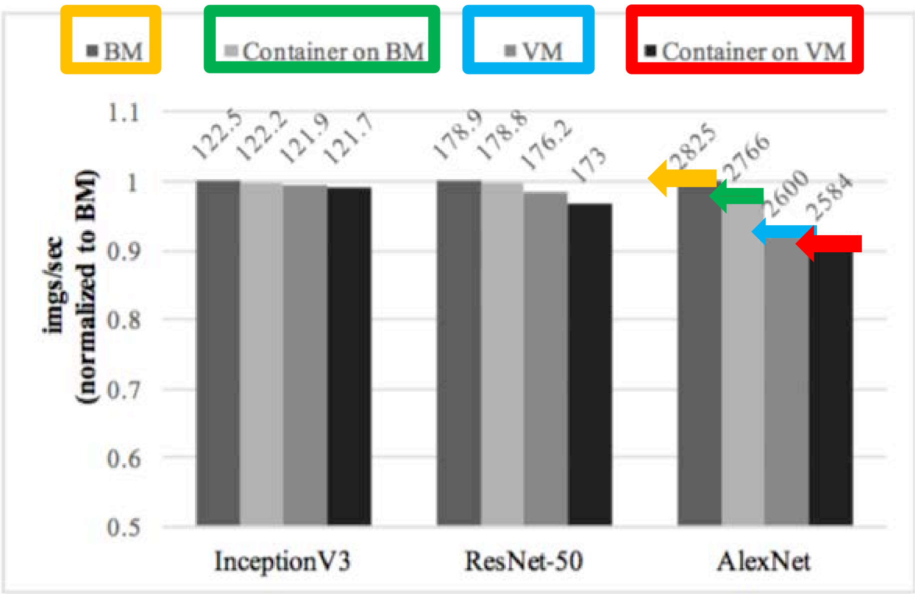


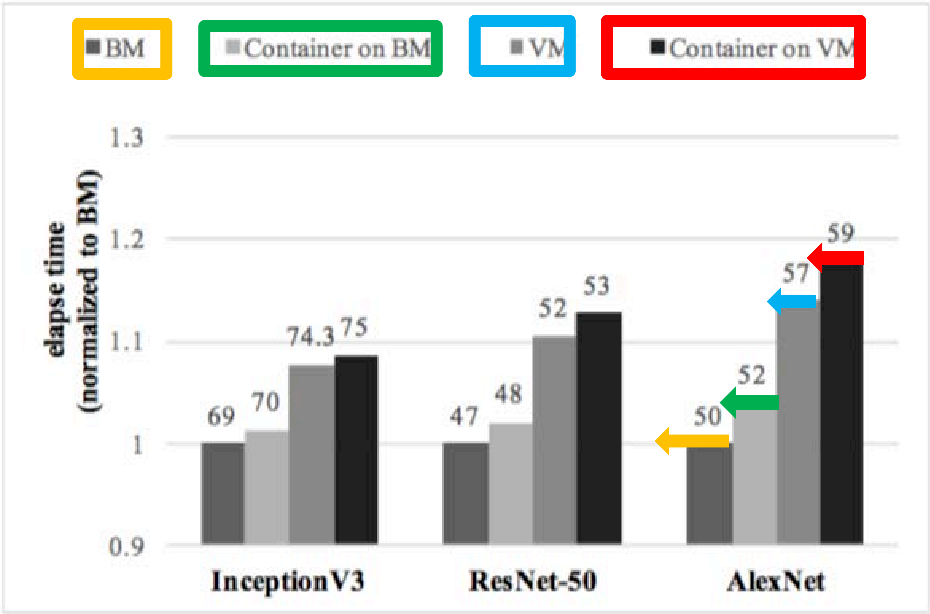
Agenda

- Introduction and Background
- Methodology
- Experiment Results
- Conclusion
- Future Work

Single Instance

- Use synthetic data (I/O is not involved)

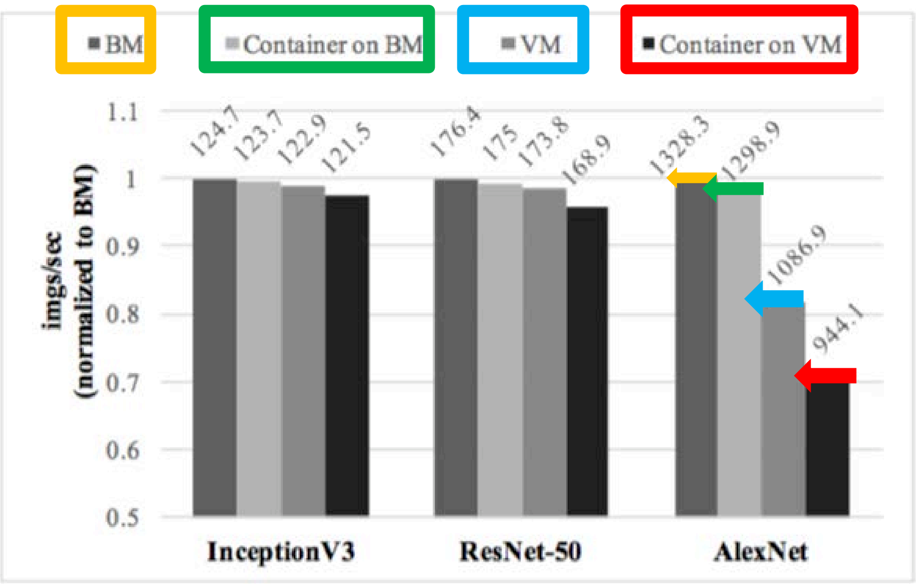


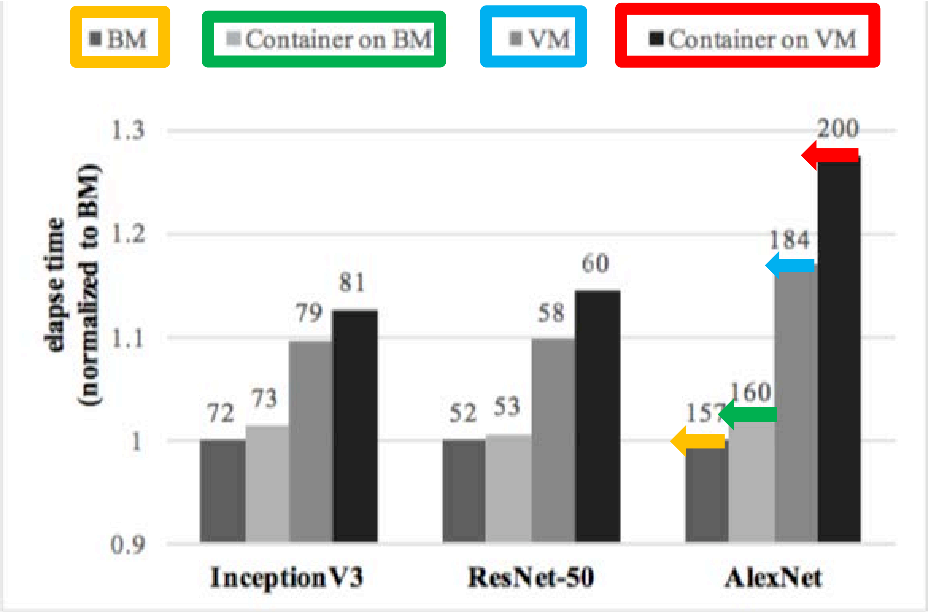


- The performance ranking is **Baremetal ≅ Container on BM > VM > Container on VM.**
- The degradation are not really big, even the performance of **Container on VM** has degradation within 20%.

Single Instance

- Use real data (I/O is involved)

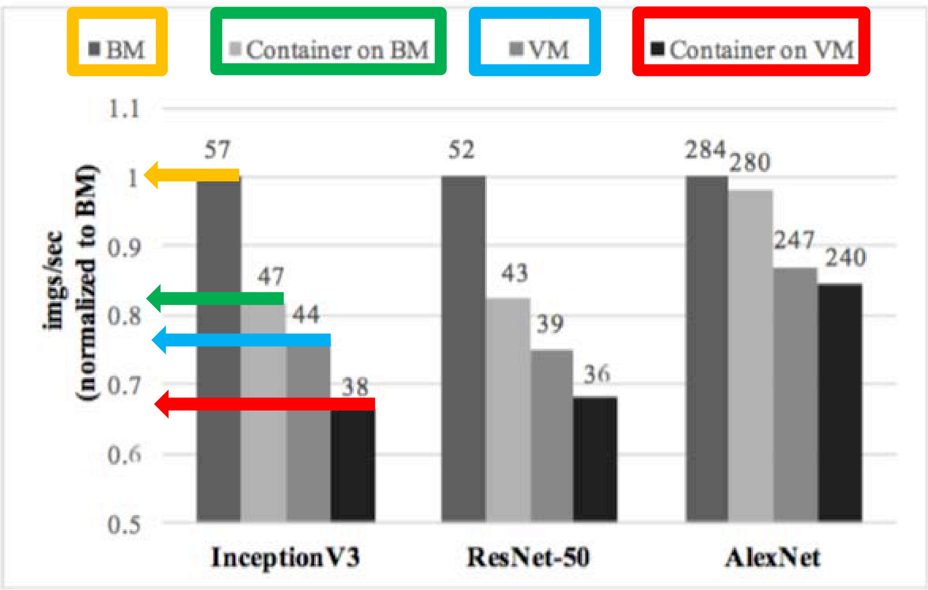


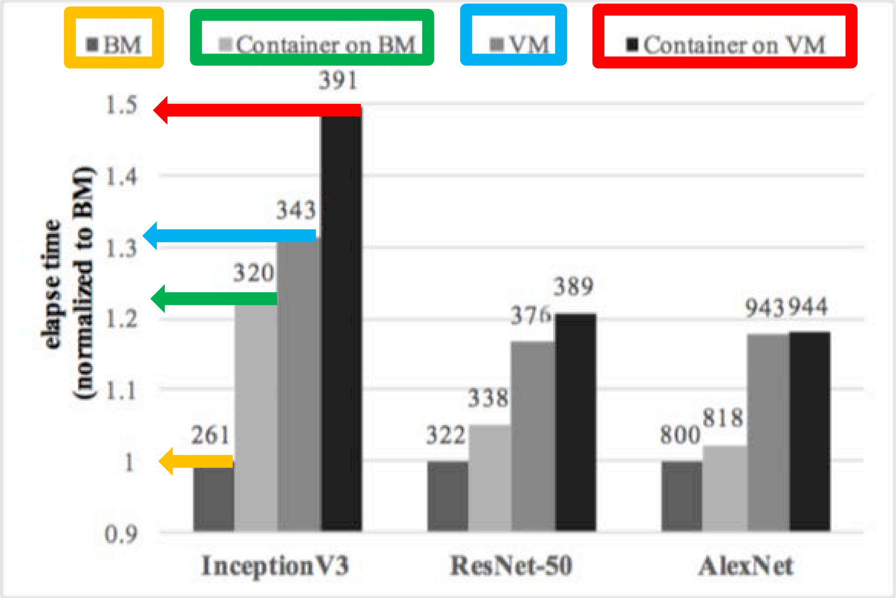


- Real data requires additional disk operations to load the training data from disk and the time is included in the elapse time.
- Compared to Baremetal, the performance of VM degrades significantly to 30%. (In synthetic data, it is 20%)
- **VM** suffers from the **I/O performance degradation**, while container on BM does not.

Distributed Multi-instances

- Use synthetic data (I/O is not involved)

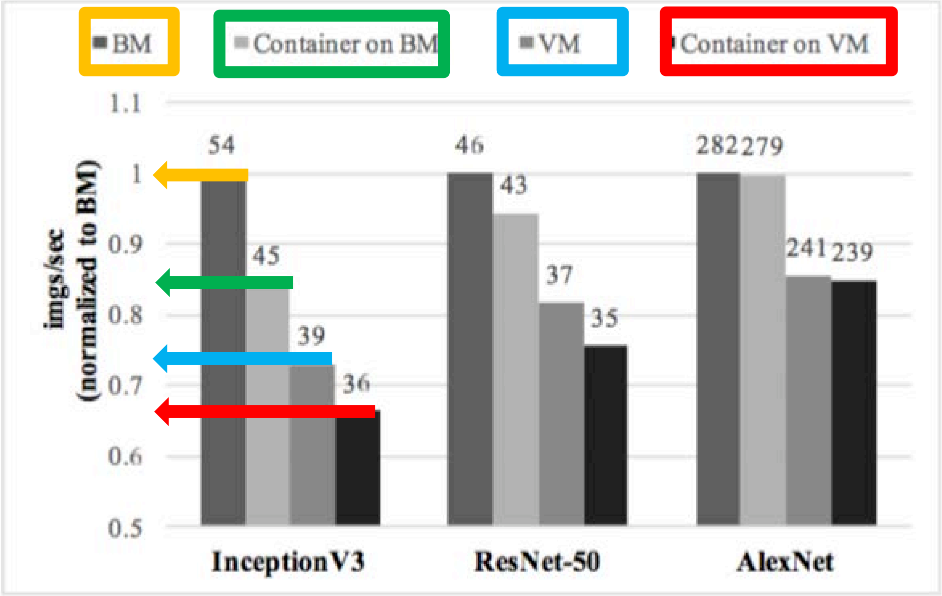


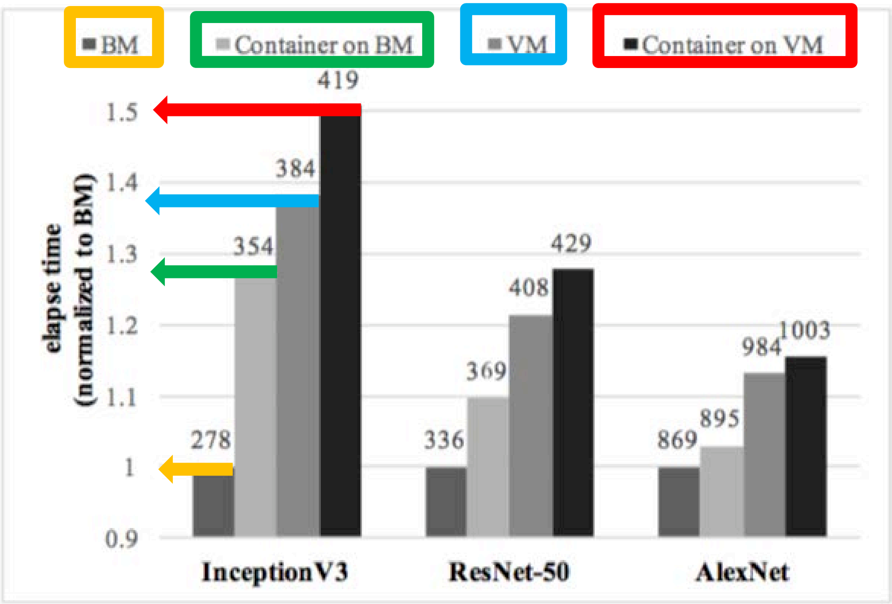


- More significant performance degradation is observed.
- **Even Container on BM surfers from performance degradation which reaches about 20%.**
- The degradation of VM and container on VM are even greater.
- I/O is not involved, network performance is the main reason.

Distributed Multi-instances

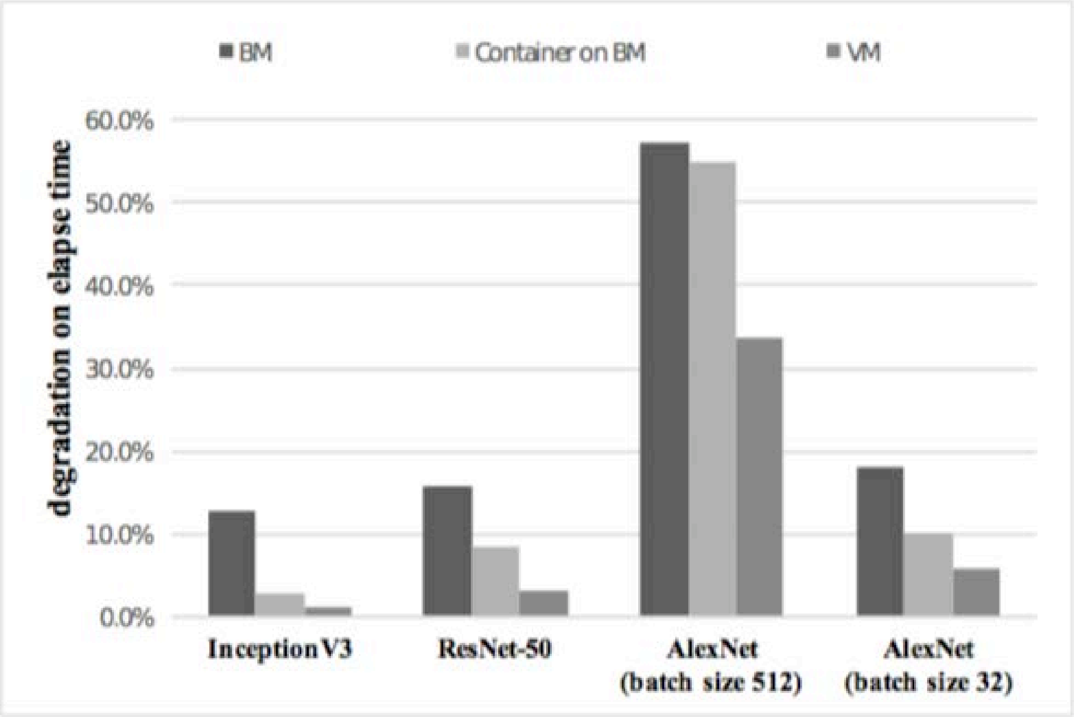
- Use real data (I/O is involved)

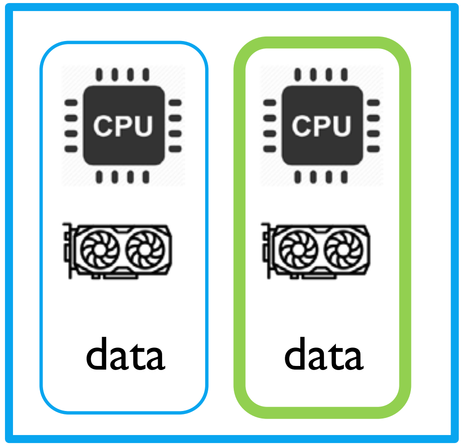




- The results of synthetic data and real data are similar. They have same level of degradation.
- Disk I/O is not a dominant reasons for the degradation.
- **Network performance** dominates the overall performance for distributed Tensorflow.
- A additional network layer, **flannel**, in the Kubernetes introduce the network overhead to Container on BM.

Shared Resource

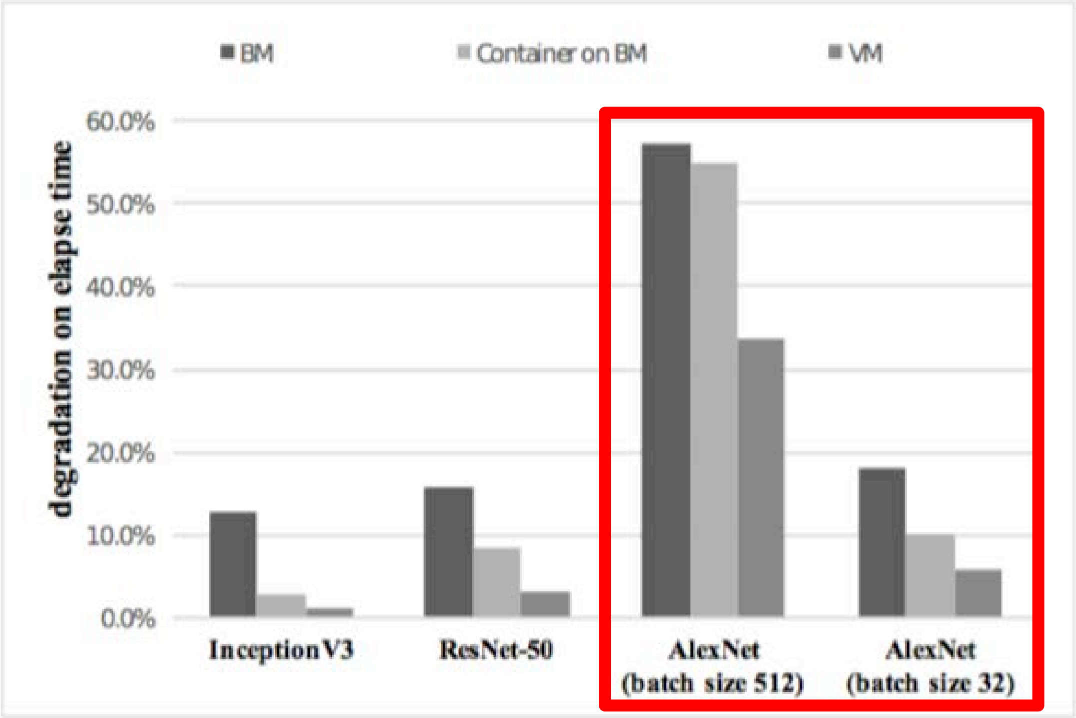


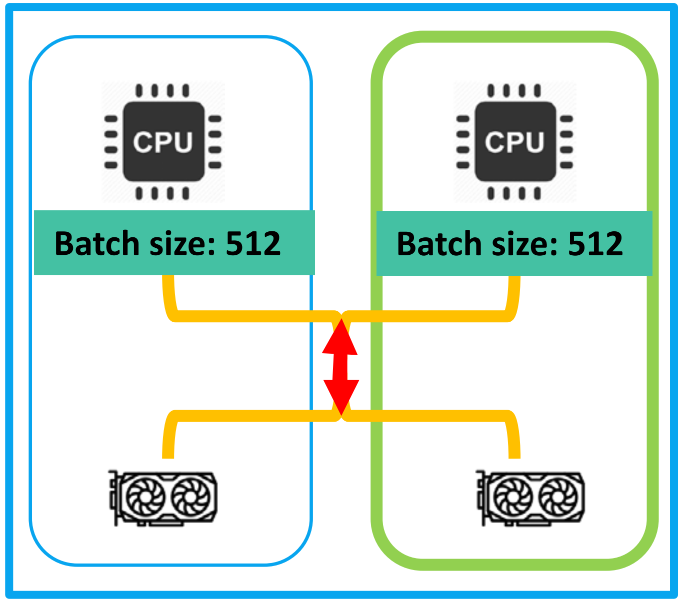


- Background workload:
  AlexNet training job with a batch size of 512.
- For baremetal:
  we directly run two training processes.
- The metrics we use is elapse time

- **Baremetal** has the highest performance impact in this environment, while **VM** has the least.
- **Containers** are bound to user designated CPUs. **Kubernetes** also has QoS mechanism to control the resource usage.
- **Virtual machine** offers the strongest resource isolation.

Shared Resource





larger batch size
=> more training data needs to be transferred.

=> Data transfer interference in I/O bus.

=> No Matter what virtualization Type.

- For **AlexNet test**, even the degradation of VM can reach 30% when batch size is set to 512.
- Reasons (from the Tensorflow trace file):
  - The execution time delay was caused by the much longer **memory copy duration between host and devices (GPU).**
  - The required bandwidth has **over the hardware I/O bus limit**, and thus cause significant performance degradation.

Agenda

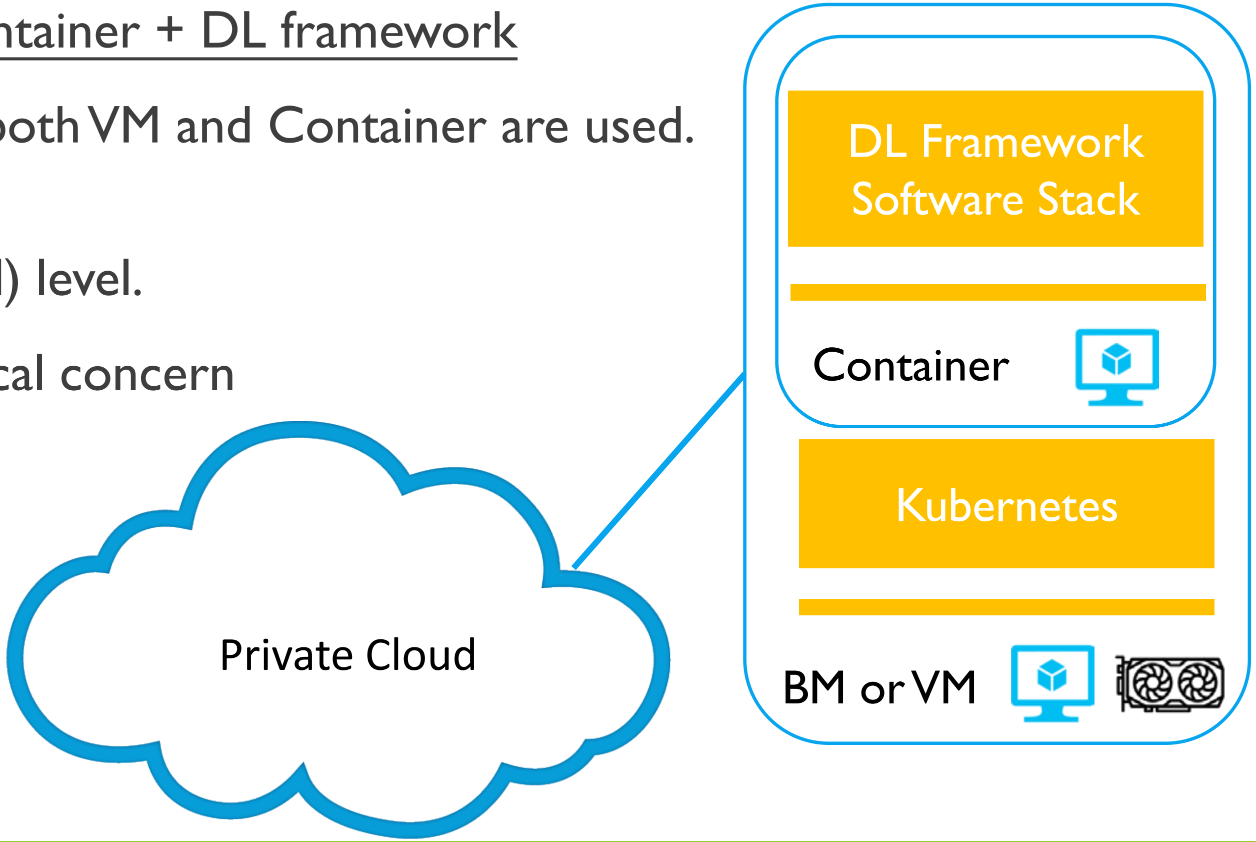
- Introduction and Background
- Methodology
- Experiment Results
- Conclusion
- Future Work

Conclusions

- For **single instance deep** learning training:

➢ Baremetal or VM + Kubernetes + Container + DL framework

1) less performance degradation even both VM and Container are used.
2) Kubernetes has fault tolerance and other functions at the container(pod) level.
3) Virtualization overhead is not a critical concern

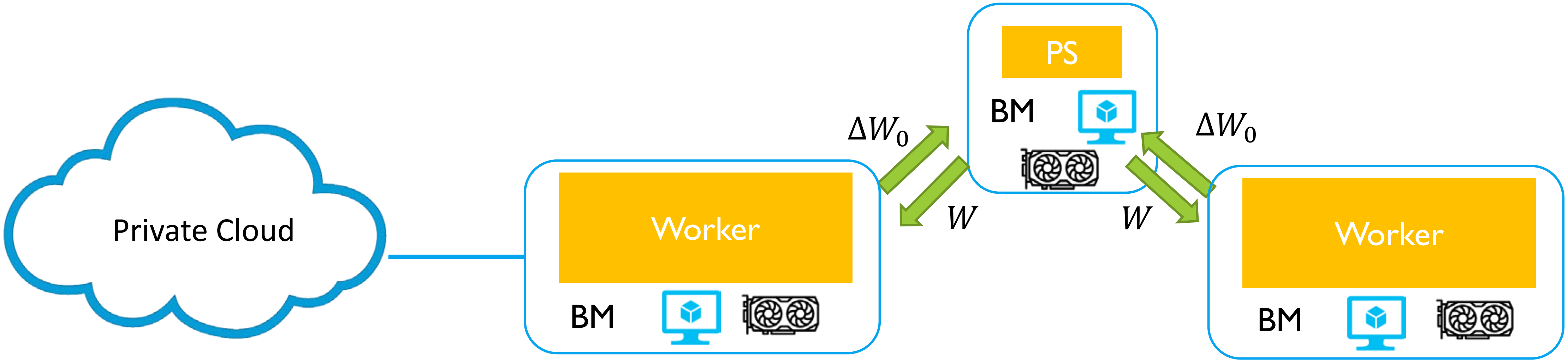


Conclusions

- For distributed multi-instances deep learning training:

➢ Baremetal + (Kubernetes + Container) + DL framework

1) Virtualization layer does cause significant degradation to network performance.
2) Kubernetes and Container needs to fix the network overlay problem,
   Ex: **Kuryr** project in OpenStack



Conclusions

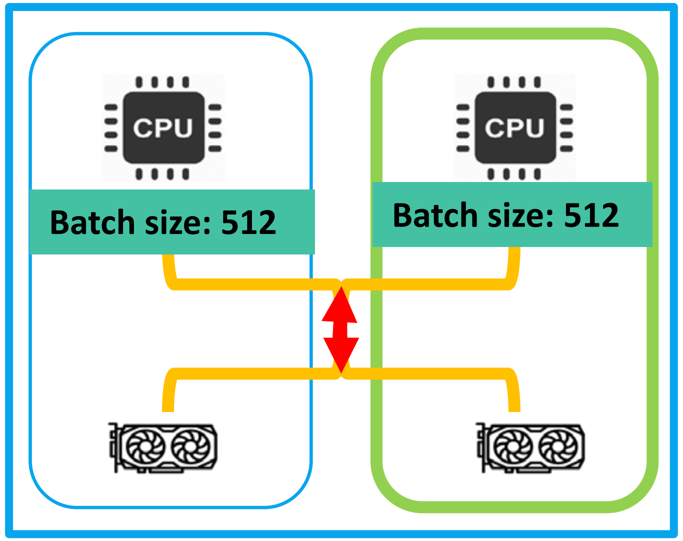
- In shared resource scenerio:

➢ Less resource contention problem since the computation is on GPU which is not shared.

➢ The resource contention on the I/O bandwidth between host and device is a major concern.

➢ No matter what type of virtualizations were used, the degradation can lead to more than 30% elapse time increment.

➢ The cloud providers need to concern the issue of **multiple training jobs on the same machine.**



Conclusions

- Performance impact of deep learning training job is **varied** according to the training model characteristics.

- If the **model** is complex (numbers of parameters are large),
  - the network overhead becomes more important.
- If the **training dataset** is large,
  - the I/O or memory access overhead becomes more critical.

Agenda

- Introduction and Background
- Methodology
- Experiment Results
- Conclusion
- Future Work

Future Work

1) Improve network virtualization performance and reduce overlay network layers in resource orchestration.

2) Provide resource sharing and controlling mechanism on a single GPU device as well as the I/O bandwidth resource between devices and hosts.

3) Develop more accurate resource usage estimation and performance prediction mechanism for deep learning job to help cloud providers optimize their job scheduling and placement decision.

Thank you!

Chan-Yi Lin, Hsin-Yu Pai and Jerry Chou
Department of Computer Science, National Tsing Hua University

March 20th, 2018
The 8th International Conference on Cloud Computing and Services Science, CLOSER 2018